Sääntöpohjainen päättely

- Sääntöpohjainen päättely:
  logiikkaohjelmointi, deduktiiviset tietokannat,
  asiantuntijajärjestelmät
- Horn-klausuulit tarjoavat loogisen pohjan sääntöpohjaiselle päättelylle.
- Horn-klausuuleille tehokkaita toteutusmenetelmiä.
- Horn-klausuulit ovat käytetyin lauselogiikan tehokkaasti ratkaistavissa oleva aliluokka.
- Tyypillisesti tarvitaan kuitenkin Horn-klausuuleja ilmaisuvoimaisempi kieli (negaatio/komplementti).

Horn-klausuulit

Määritelmä. Horn-klausuuli on literaalien disjunktio, jossa on korkeintaan *yksi* positiivinen literaali.

Esimerkki. $\neg P \vee \neg Q \vee \neg R$ ja $\neg P \vee Q \vee \neg R$ ovat Horn-klausuuleja.

Esimerkki. Klausuuli $P \vee \neg Q \vee R$ ei ole Horn-klausuuli.

Merkintätapa.
Horn-klausuuli $\neg P_1 \vee \cdots \vee \neg P_n \vee Q$ kirjoitetaan usein muodossa

$$Q \leftarrow P_1 \wedge \cdots \wedge P_n$$

ja Horn-klausuuli $\neg P_1 \vee \cdots \vee \neg P_n$ muodossa

$$\leftarrow P_1 \wedge \cdots \wedge P_n$$

Ohjelmaklausuulit

Tyypillisesti käytetään ohjelmaklausuuleja
(engl. program clauses; definite clauses).

Määritelmä. *Ohjelmaklausuuli* on literaalien disjunktio $A \vee \neg B_1 \vee \cdots \vee \neg B_n$, jossa on täsmälleen *yksi* positiivinen literaali.

Esimerkki. Klausuuli $\neg P \vee \neg Q \vee \neg R$ ei ole ohjelmaklausuuli, mutta $\neg P \vee Q \vee \neg R$ (eli $Q \leftarrow P \wedge R$) on.

Väite. Ohjelmaklausuulijoukko on toteutuva.

Ohjelmaklausuulijoukolle saadaan malli M esim. merkitsemällä jokaisen klausuulin $A \vee \neg B_1 \vee \cdots \vee \neg B_n$ positiivinen atomi A todeksi.

Huomio. Horn-klausuulien joukko ei ole välttämättä toteutuva. Esimerkiksi $\{P, \neg P\}$ on toteutumaton.

- Merkitään jatkossa ohjelmaklausuulien joukossa S esiintyvien atomien joukkoa $\mathrm{HB}(S)$:llä (kysymyksessä Herbrand-kanta).
- Horn-klausuulien joukolle S voidaan määritellä vastaava ohjelmaklausuulijoukko S' ottamalla käyttöön uusi atomi $\bot$ ja kirjoittamalla S:n puhtaasti negatiiviset Horn-klausuulit

  $$\neg P_1 \vee \cdots \vee \neg P_n$$

  muotoon

  $$\bot \vee \neg P_1 \vee \cdots \vee \neg P_n \text{ (eli } \bot \leftarrow P_1 \wedge \cdots \wedge P_n).$$

☞ Horn-klausuulien toteutuvuusongelma voidaan ratkaista ohjelmaklausuuliesityksen avulla.

Väite. Joukko Horn-klausuuleja S on toteutuva $\iff S' \not\models \bot$, missä S' on joukkoa S vastaava ohjelmaklausuulien joukko.

Esimerkki. $\{P, \neg P\}$ ei ole toteutuva ja $\{P, \bot \leftarrow P\} \models \bot$.

Ohjelmaklausuulien minimimalli

Määritelmä. (i) Malli M_1 (atomilauseiden joukko) on pienempi kuin malli M_2 ($M_1 < M_2$), joss $M_1 \subset M_2$.

(ii) Klausuulijoukon mallia M (atomilauseiden joukko) sanotaan minimaaliseksi, jos lausejoukolla ei ole sitä pienempää mallia.

Esimerkki. Tarkastellaan ohjelmaklausuulien joukkoa

$$S = \{Q \leftarrow R, R \leftarrow P \wedge Q\}.$$

- Totuusjakelu $M = \{Q, R\}$ on S:n malli.
- M ei ole minimaalinen, koska $M' = \emptyset$ on myös S:n malli.
- Sen sijaan M' on S:n minimimalli.

Teoreema. Olkoon $M_i \subseteq \mathrm{HB}(S)$ (missä $i \in I$) mikä tahansa kokoelma ohjelmaklausuulien joukon S malleja. Tällöin myös leikkaus

$$M = \bigcap\{M_i \mid i \in I\}$$

on malli joukolle S.

Todistus. Tehdään vastaoletus $M \not\models S$.

$\Longrightarrow$ $\exists A \leftarrow B_1 \wedge \cdots \wedge B_n \in S$ s.e. $\{B_1, \ldots, B_n\} \subseteq M$, mutta $A \notin M$

$\Longrightarrow$ kaikille $i \in I$ pätee $\{B_1, \ldots, B_n\} \subseteq M_i$

$\Longrightarrow$ kaikille $i \in I$ pätee $A \in M_i$, koska $M_i \models S$, $A \leftarrow B_1 \wedge \cdots \wedge B_n \in S$ ja $M_i \models A \leftarrow B_1 \wedge \cdots \wedge B_n$

$\Longrightarrow$ $A \in M = \bigcap\{M_i \mid i \in I\}$, ristiriita. $\square$

Teoreema. Ohjelmaklausuulijoukolla S on ainakin yksi minimimalli.

Todistus. Koska S on toteutuva, sillä on ainakin yksi malli M_0.

Määritellään laskeva mallien sekvenssi $M_0, M_1, \ldots$ seuraavasti:

1. Seuraajaordinaaleille α:
   - Jos $M_{\alpha-1}$ on S:n minimimalli, määritellään $M_\alpha = M_{\alpha-1}$
   - Jos $M_{\alpha-1}$ ei ole S:n minimimalli, on S:llä malli $M \subset M_{\alpha-1}$, jolloin määritellään $M_\alpha = M$.
2. Rajaordinaaleille α:
   - Määritellään $M_\alpha = \bigcap_{\beta<\alpha} M_\beta$ (joka on myös S:n mallien leikkauksena S:n malli).

Jos sekvenssi oletetaan aidosti laskevaksi päädytään ristiriitaan, kun α:n kardinaliteetti ylittää joukon $\mathrm{HB}(S)$ kardinaliteetin
$\Longrightarrow$ S:lle saadaan minimimalli jollakin ordinaalilla α.

Teoreema. Joukolla ohjelmaklausuuleja S on yksikäsitteinen minimimalli M_S (pienin malli), joka on S:n kaikkien mallien leikkaus.

Todistus. Edellä osoitettiin, että S:llä on ainakin yksi minimimalli.

Oletetaan, että M_1 ja M_2 ovat S:n minimimalleja.

$\Longrightarrow$ $M_1 \cap M_2$ on S:n malli.

$\Longrightarrow$ $M_1 \cap M_2 = M_1$ ja $M_1 \cap M_2 = M_2$, koska M_1 ja M_2 ovat minimimalleja

$\Longrightarrow$ $M_1 = M_2$.

- Kaikille S:n malleille $M \subseteq \mathrm{HB}(S)$ pätee $M_S \subseteq M$, koska M_S osoitettiin edellä yksikäsitteiseksi.
- Täten kaikkien S:n mallien (joihin myös M_S lukeutuu) leikkaus $\bigcap\{M \subseteq \mathrm{HB}(S) \mid M \models S\}$ on täsmälleen M_S. □

Teoreema. Olkoon S joukko ohjelmaklausuuleja. Tällöin

$$M_S = \{P \in \mathrm{HB}(S) \mid S \models P\}.$$

Todistus. Tarkastellaan mitä hyvänsä atomia $P \in \mathrm{HB}(P)$. Nyt

$$\begin{aligned} S \models P \quad &\iff \quad M \models P \text{ kaikille } S\text{:n malleille } M \subseteq \mathrm{HB}(S) \\ &\iff \quad P \in M \text{ kaikille } S\text{:n malleille } M \subseteq \mathrm{HB}(S) \\ &\iff \quad P \in M_S, \end{aligned}$$

koska $M_S = \bigcap\{M \subseteq \mathrm{HB}(S) \mid M \models S\}$.

Minimimallin konstruointi

Määritelmä. Olkoon S (mahd. ääretön) joukko ohjelmaklausuuleja. Määritellään operaattori $\mathrm{T}_S : \mathbf{2}^{\mathrm{HB}(S)} \to \mathbf{2}^{\mathrm{HB}(S)}$ seuraavasti:

$$\mathrm{T}_S(M) = \{P \mid P \leftarrow P_1 \wedge \cdots \wedge P_n \in S \text{ ja } \{P_1, \ldots, P_n\} \subseteq M\}$$

Esimerkki. Olkoon $S = \{P \leftarrow P, Q, R_1 \leftarrow Q, R_2 \leftarrow Q \wedge P\}$.

Nyt $\mathrm{T}_S(\{P\}) = \{P, Q\}$ ja $\mathrm{T}_S(\{P, Q\}) = \{P, Q, R_1, R_2\}$.

- M on kuvauksen T_S *kiintopiste*, joss $\mathrm{T}_S(M) = M$.

  Esimerkki. $M_1 = \{P, Q, R_1, R_2\}$ on kuvauksen T_S kiintopiste: $\mathrm{T}_S(M_1) = \{P, Q, R_1, R_2\} = M_1$

- Kiintopiste M on pienin, jos muille kiintopisteille M', $M \subseteq M'$.

  Esimerkki. $M_2 = \{Q, R_1\}$ on kuvauksen T_S pienin kiintopiste.

- Operaattori T_S on monotoninen:

  kaikille $M \subseteq M' \subseteq \mathrm{HB}(S)$ pätee $\mathrm{T}_S(M) \subseteq \mathrm{T}_S(M')$.

- Knaster-Tarski -teoreema: jokaisella monotonisella operaattorilla on yksikäsitteinen pienin (suurin) kiintopiste.

Teoreema. Operaattorilla T_S on pienin kiintopiste

$$\mathrm{lfp}(\mathrm{T}_S) = \bigcap\{M \subseteq \mathrm{HB}(S) \mid \mathrm{T}_S(M) \subseteq M\}.$$

Väite. Atomilauseiden joukko $M \subseteq \mathrm{HB}(S)$ on ohjelmaklausuulijoukon S malli, joss $\mathrm{T}_S(M) \subseteq M$.

Todistus. $M \not\models S$
$\iff$ $\exists A \leftarrow B_1 \wedge \cdots \wedge B_n \in S$ s.e. $\{B_1, \ldots, B_n\} \subseteq M$, mutta $A \notin M$
$\iff$ $\exists A \in \mathrm{T}_S(M)$ s.e. $A \notin M$ $\iff$ $\mathrm{T}_S(M) \not\subseteq M$. $\square$

Teoreema. Pienin malli $M_S = \mathrm{lfp}(\mathrm{T}_S)$.

Määritelmä.
$\mathrm{T}_S \uparrow 0 = \emptyset$
$\mathrm{T}_S \uparrow \alpha = \mathrm{T}_S(\mathrm{T}_S \uparrow \alpha - 1)$
$\mathrm{T}_S \uparrow \omega = \bigcup\{\mathrm{T}_S \uparrow \alpha \mid \alpha < \omega\}$

Teoreema. Pienin kiintopiste $\mathrm{lfp}(\mathrm{T}_S) = \mathrm{T}_S \uparrow \omega$.

Esimerkki. Joukolle $S = \{P \leftarrow P,\ Q,\ R_1 \leftarrow Q,\ R_2 \leftarrow Q \wedge P\}$:

$$\begin{aligned}
&\mathrm{T}_S \uparrow 0 = \emptyset,\\
&\mathrm{T}_S \uparrow 1 = \mathrm{T}_S(\mathrm{T}_S \uparrow 0) = \mathrm{T}_S(\emptyset) = \{Q\},\\
&\mathrm{T}_S \uparrow 2 = \mathrm{T}_S(\mathrm{T}_S \uparrow 1) = \mathrm{T}_S(\{Q\}) = \{Q, R_1\},\\
&\mathrm{T}_S \uparrow 3 = \mathrm{T}_S(\mathrm{T}_S \uparrow 2) = \mathrm{T}_S(\{Q, R_1\}) = \{Q, R_1\},\\
&\vdots\\
&\mathrm{T}_S \uparrow \omega = \bigcup\{\mathrm{T}_S \uparrow \alpha \mid \alpha < \omega\} = \{Q, R_1\}.
\end{aligned}$$

Johdettavuus ohjelmalauseilla

Määritelmä. Atomilauseen P on johto ohjelmalauseista S on äärellinen jono atomilauseita $P_0, \ldots, P_n$ siten, että $P = P_n$ ja

(i) $P_0 \in S$ ja

(ii) jokaiselle $i > 0$ on olemassa $P_i \leftarrow Q_1 \wedge \cdots \wedge Q_n \in S$ siten, että $\{Q_1, \ldots, Q_n\} \subseteq \{P_1, \ldots, P_{i-1}\}$.

Jos lauseelle P on olemassa johto joukosta S, merkitään tätä $S \vdash P$.

Teoreema. (i) $S \models P \iff S \vdash P$ ja
(ii) $M_S = \{P \mid P$ on atomilause ja $S \vdash P\}$.

Esimerkki. $S = \{P \leftarrow P,\ Q,\ R_1 \leftarrow Q,\ R_2 \leftarrow Q \wedge R_1,\ R_2 \leftarrow P\}$.

Jono Q, R_1, R_2 on lauseen R_2 johto joukosta S ($S \vdash R_2$).

Jono Q, P, R_2 ei ole lauseen R_2 johto joukosta S.

Logiikkaohjelmat ja deduktiiviset tietokannat

- Logiikkaohjelmat (ilman ei-loogisia operaatioita: cut, not, assert, retract) ja deduktiiviset tietokannat voidaan tulkita ohjelmaklausuulien joukoiksi S, joissa literaalit sisältävät muuttujia sekä vakio- ja funktiosymboleja.
- Säännöt ovat siis muotoa

  $$P(\vec{t}) \leftarrow Q_1(\vec{t_1}) \wedge \cdots \wedge Q_n(\vec{t_n})$$

  missä $\vec{t}$, $\vec{t_1}$, ..., $\vec{t_n}$ ovat predikaattien P, P_1, ..., P_n argumentteina olevia termilistoja.
- Mikä on tällaisen ohjelmaklausuulijoukon S semantiikka: milloin annettuun kyselyyn $Q(\vec{t})$ pitää antaa myönteinen vastaus?

- Jokaiseen ohjelmaklausuulien joukkoon S liittyy Herbrand-universumi $\mathrm{HU}(S)$ eli S:n vakio- ja funktiosymboleista muodostettavissa olevien muuttujattomien termien joukko.
- Mikä tahansa S:n muuttujia sisältävä ohjelmaklausuuli $P(\vec{t}) \leftarrow Q_1(\vec{t_1}) \wedge \cdots \wedge Q_n(\vec{t_n})$ voidaan *instatioida* korvaamalla muuttujat $\mathrm{HU}(S)$: muuttujattomien termien eri kombinaatioilla.
- Muuttujia sisältävä ohjelmaklausuulien joukko S edustaa *Herbrand-instanssiensa* joukkoa S_{H}, joka voidaan tulkita propositionaaliseksi ohjelmaklausuulien joukoksi.
- Joukolla S_{H} on yksikäsitteinen pienin Herbrand malli $M_{S_{\mathrm{H}}}$.
- Tästä saadaan kriteeri kyselyjen oikeellisuudelle: olkoon $x_1, \ldots, x_n$ kyselyssä $Q(\vec{t})$ esiintyvät muuttujat.
  Nyt $S \models \exists x_1 \cdots \exists x_n Q(\vec{t}) \iff$ on olemassa vastaussubstituutio θ (korvaa x_i:t $\mathrm{HU}(S)$:n termeillä) s.e. $Q(\vec{t})\theta \in M_{S_{\mathrm{H}}}$.

Esimerkki. Deduktiivinen tietokanta S:

```
p(a,c).  p(b,c).  q(X) :- p(X,Y).
```

- Herbrand-unversumi $\mathrm{HU}(S) = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$.
- Herbrand-instanssien joukko S_{H}:

```
p(a,c). p(b,c).
q(a) :- p(a,a).  q(a) :- p(a,b).  q(a) :- p(a,c).
q(b) :- p(b,a).  q(b) :- p(b,b).  q(b) :- p(b,c).
q(c) :- p(c,a).  q(c) :- p(c,b).  q(c) :- p(c,c).
```

- Pienin Herbrand-malli $M_{S_{\mathrm{H}}} = \{$`p(a,c), p(b,c), q(a), q(b)`$\}$.
- $M_{S_{\mathrm{H}}}$ antaa kyselyille odotetut vastaukset:

```
?- p(a,c). kyllä        ?- p(a,d). ei
?- q(a). kyllä          ?- q(c). ei
```

Esimerkki. Olkoon tietokanta S seuraava:

```
vanhempi(kaija,jussi). vanhempi(tauno,jussi).
vanhempi(kaija,teija). vanhempi(saija,kaija).
vanhempi(raija,saija).
sisarus(X,Y) :- vanhempi(Z,X), vanhempi(Z,Y).
esivanhempi(X,Y) :- vanhempi(X,Y).
esivanhempi(X,Y) :- vanhempi(X,Z), esivanhempi(Z,Y).
```

➤ Pienin Herbrand-malli $M_{S_{\mathrm{H}}}$ antaa odotetut vastaukset:

```
?- vanhempi(kaija,jussi). kyllä      ?- vanhempi(raija,jussi). ei
?- sisarus(teija,jussi). kyllä       ?- sisarus(teija,jukka). ei
?- esivanhempi(raija,jussi). kyllä   ?- esivanhempi(raija,jari). ei
```


Sääntökielen ilmaisuvoima

Säännöillä pystytään ilmaisemaan keskeiset relaatioalgebran operaatiot:

1. Unioni: $P(x_1, \ldots, x_n) \leftarrow Q_1(x_1, \ldots, x_n)$
   $P(x_1, \ldots, x_n) \leftarrow Q_2(x_1, \ldots, x_n)$
2. Leikkaus: $P(x_1, \ldots, x_n) \leftarrow Q_1(x_1, \ldots, x_n) \wedge Q_2(x_1, \ldots, x_n)$
3. Projektio: $\mathrm{Vanhempi}(x) \leftarrow \mathrm{Vanhempi}(x, y)$
4. Valinta: $\mathrm{Miljonaari}(x) \leftarrow \mathrm{Tulot}(x, y) \wedge (y > 1000000)$
5. Kompositio: $\mathrm{Tenttitulos}(x, y) \leftarrow \mathrm{Opiskelija}(x, i) \wedge \mathrm{Arvosana}(i, y)$

- Verrattuna relaatioalgebraan (SQL): komplementti puuttuu.
- Toisaalta säännöt ovat ilmaisuvoimaisempia kuin relaatioalgebra (SQL): relaatioiden määritelmät voivat olla rekursiivisia.

  Esimerkki. Transitiivinen sulkeuma:

  $\mathrm{Yhteys}(x, y) \leftarrow \mathrm{Lento}(x, y)$
  $\mathrm{Yhteys}(x, y) \leftarrow \mathrm{Lento}(x, y) \wedge \mathrm{Yhteys}(y, z)$
- Tyypillisesti lisätään komplementti jossain muodossa.

  Esimerkki. Ad hoc -ratkaisut johtavat ongelmiin:

  $\mathrm{Oluenjuoja}(x) \leftarrow \mathrm{DI}(x) \wedge \mathrm{not}\ \mathrm{Absolutisti}(x),$
  $\mathrm{Absolutisti}(x) \leftarrow \mathrm{Ekonomi}(x) \wedge \mathrm{not}\ \mathrm{Oluenjuoja}(x),$
  $\mathrm{DI}(\mathrm{Liisa}),\ \mathrm{Ekonomi}(\mathrm{Liisa})$

Toteutustekniikat

- Prologin toteutus ei ole täydellinen.

  Esimerkki. Ohjelma `p :- p.` ja kysely `?- p.`

  Vastauksen pitäisi olla kieltävä mutta Prolog jää silmukkaan.
- Bottom-up: magic sets

  Deduktiiviset tietokannat
- Top-down: SLG-resoluutio (tabling-tekniikka)

  XSB-Prolog
- Muuttujattomalle ohjelmalle minimimalli voidaan laskea lineaarisessa ajassa [Dowling-Gallier, 1984].

Ratkaisumenetelmä prop. Horn-klausuuleille

- Horn-klausuulit muodostavat keskeisimmän tehokkaasti toteutettavissa oleva osajoukon lauselogiikasta.
- Ratkaisuongelma: onko äärellinen joukko propositionaalisia Horn-klausuuleja S toteutuva?
- Kun tämä osataan ratkaista tehokkaasti, saamme myös tehokkaan tavan ratkaista ongelman

$$S \models Q_1 \vee \cdots \vee Q_n$$

missä S on Horn-klausuulijoukko ja jokainen Q_i on literaalien konjunktio, jossa esiintyy korkeintaan yksi negatiivinen literaali.

Huomio. $S \models Q \iff S \cup \{\neg Q\}$ on toteutumaton.

Neliöllinen ratkaisumenetelmä

1. Joukko Horn-klausuuleja S on toteutuva $\iff S' \not\models \bot$ (S' on joukkoa S vastaava ohjelmaklausuulien joukko).
2. $S' \not\models \bot \iff \bot \notin M_{S'}$.
3. Malli $M_{S'}$ voidaan laskea operaattorilla $\mathrm{T}_{S'}$:

$$\mathrm{T}_{S'} \uparrow 0,\ \mathrm{T}_{S'} \uparrow 1,\ \mathrm{T}_{S'} \uparrow 2,\ \ldots$$

kunnes saavutetaan kiintopiste ($\mathrm{T}_{S'} \uparrow n = \mathrm{T}_{S'} \uparrow n-1$).

4. Iteraatioita tarvitaan korkeintaan $|\mathrm{HB}(S)| + 1$ kappaletta.
5. Joukon $\mathrm{T}_{S'}(M)$ laskenta voidaan suorittaa mille tahansa $M \subseteq \mathrm{HB}(S')$ lineaarisessa ajassa $||S'||$:n suhteen.
6. Aikakompleksisuus: $\mathcal{O}(|\mathrm{HB}(S')| \times ||S'||)$.

Lineaarinen ratkaisumenetelmä

- W.F. Dowling ja J.H. Gallier [1984]: *Linear-Time Algorithms for Testing the Satisfiability of Propositional Horn Formulae*
- Idea: toteutetaan T_S-iteraatio tehokkaalla tietorakenteella siten, että kutakin atomia käsitellään vain kerran.
- Käytetään taulukoita, joita varten tarvitaan kullekin atomille yksikäsitteinen indeksi $\mathrm{HB}(S) = \{P_0, P_1, \ldots, P_{n-1}\}$.
- Jos indeksejä ei tiedetä etukäteen ja syötejoukko joudutaan jäsentämään, vaatii indeksien muodostaminen $\mathcal{O}(||S|| \times \log |\mathrm{HB}(S)|)$ työn (symbolitaulu).
- Tämän jälkeen algoritmi toimii lineaarisessa ajassa $||S||$:n suhteen.

Tietorakenteet

Olkoon syötejoukossa n atomilausetta ja m ohjelmaklausuulia.

Otetaan käyttöön seuraavat taulukot:

- val[0 ... n-1]: atomien $P_0, P_1, \ldots, P_{n-1}$ totuusarvot (aluksi true, jos P_i esiintyy ainoana positiivisena literaalina jossain ohjelmaklausuulissa, muutoin false).
- occurs[0 ... n-1]: listat ohjelmaklausuuleista joissa atomit $P_0, P_1, \ldots, P_{n-1}$ esiintyvät negatiivisina.
- negcnt[0 ... m-1]: klausuulien negatiivisten literaalien lukumäärät.
- poslit[0 ... m-1]: klausuulien positiiviset literaalit.

Algoritmi

Luetaan syötejoukko;
Alustetaan taulukot val[0 ... n-1], occurs[0 ... n-1],
  negcnt[0 ... m-1] ja poslit[0 ... m-1];
consistent := true;
falseatom := atomin $\perp$ indeksi;
queue := atomit, joille val[i] aluksi true;
minmodel;
/* Laskee syötteenä annetun ohjelmaklausuulijoukon pienimmän mallin taulukkoon val paitsi, jos $\perp$ saa arvon tosi, jolloin keskeytetään. */
if consistent **then**
  print('satisfying model: '); printmodel
else print('unsat set of clauses')
endif

```
procedure minmodel;
while not empty(queue) and consistent do
  a := get(queue);
  clauses := occurs[a] ;
  while not empty(clauses) and consistent do
    clause := get(clauses); negcnt[clause] := negcnt[clause]-1;
    if negcnt[clause]=0 then
      n := poslit[clause];
      if val[n]=false then
        val[n] := true;
        if n=falseatom then consistent := false else push(n, queue);
      endif
    endif
  done
done
```

Esimerkki. Olkoon $S =$

$\{0 : P_0,\ 1 : P_1 \leftarrow P_0,\ 2 : P_2 \leftarrow P_1 \wedge P_0,\ 3 : P_3 \leftarrow P_1,\ 4 : \bot_5 \leftarrow P_2 \wedge P_4\}.$

Algoritmi laskee mallin $M_S = \{P_0, P_1, P_2, P_3\}$ seuraavasti:

```
val[0] := true
queue := ⟨0⟩
-----------------
a := 0
clauses := [1,2]
  clause := 1
  negcnt[1] := 0
  val[1] := true
  queue := ⟨1⟩
  clause := 2
  negcnt[2] := 1
```

```
a := 1
clauses := [2,3]
  clause := 2
  negcnt[2] := 0
  val[2] := true
  queue := ⟨2⟩
  clause := 3
  negcnt[3] := 0
  val[3] := true
  queue := ⟨2,3⟩
```

```
a := 2
clauses := [4]
  clause := 4
  negcnt[4] := 1
-----------------
a := 3
clauses := []
```


Huomioita Dowling-Gallier-algoritmista

- Oikeellisuus: val[i]=true $\iff$ $P_i \in \mathrm{lfp}(\mathrm{T}_S)$.
- Lineaarinen aikavaativuus suhteessa $||S||$:een.
  1. Kukin atomi päätyy jonoon (queue) korkeintaan kerran.
  2. Sisemmässä **while**-silmukassa tehdään työtä kokonaisuudessaan korkeintaan negatiivisten literaalien esiintymien lukumäärän verran.
- Algoritmin suorittaman hakutyyppi voidaan valita:
  - Jono $\Rightarrow$ leveyshaku
  - Pino $\Rightarrow$ syvyyshaku
- Samaa ideaa voidaan käyttää yksikköresoluution toteuttamiseen lineaarisessa ajassa (esim. Davis-Putnam -menetelmässä).